```
MODULE Main ;

IMPORT IO, Tick ;
FROM Aufgabe35 IMPORT Random ;


(* Verbessertes QuickSort: Laesst Teilfolgen mit bis zu k Elementen
   unsortiert.  Verbliebene Unordnung wird mit InsertionSort bereinigt. *)


PROCEDURE QuickSort(VAR a : ARRAY OF INTEGER ; k : CARDINAL) =

    PROCEDURE Partition(l, r : INTEGER) : INTEGER =
    VAR v, t, i, j : INTEGER ;
    BEGIN
        i := l - 1 ;
        j := r ;
        v := a[r] ; (* waehle Pivot-Element *)
        REPEAT
            REPEAT INC(i) UNTIL a[i] >= v ;
            REPEAT DEC(j) UNTIL a[j] <= v ;
            t := a[i] ; a[i] := a[j] ; a[j] := t ;
        UNTIL j <= i ; (* Zeiger kreuzen *)
        a[j] := a[i] ; a[i] := a[r] ; a[r] := t ;
        RETURN i ;
    END Partition ;

    PROCEDURE QS(l, r : INTEGER) =
    VAR i : INTEGER ;
    BEGIN
        IF r > l + k THEN (* <- einziger Unterschied zum normalen QuickSort *)
            i := Partition(l, r) ;
            QS(l, i-1) ;
            QS(i+1, r) ;
        END ;
    END QS ;

BEGIN
    QS(1, LAST(a)) ;
END QuickSort ;


PROCEDURE InsertionSort(VAR a : ARRAY OF INTEGER) =
VAR j, t : INTEGER ;
BEGIN
    FOR i := 1 TO LAST(a) DO
        t := a[i] ;
        j := i ;
        WHILE t < a[j-1] DO
            a[j] := a[j-1] ;
            DEC(j) ;
        END ;
        a[j] := t ;
    END ;
END InsertionSort ;


PROCEDURE CombinedSearch(VAR a : ARRAY OF INTEGER ; k : CARDINAL) =
BEGIN
    a[0] := 0 ; (* Sentinel *)
    QuickSort(a, k) ;
    InsertionSort(a) ;
END CombinedSearch ;


```



```
(* Hauptprogramm fuehrt einige Laufzeitmessungen zur Bestimmung der
   optimalen Konstanten k durch *)

CONST n = 1000000 ;
      k_max = 200 ;
      num_tests = 10 ;

VAR data, work : ARRAY [0..n] OF INTEGER ;
    t1, t2 : Tick.T ;
    times := ARRAY [0..k_max] OF REAL {0.0,..} ;

BEGIN
    FOR t := 1 TO num_tests DO
        (* Feld mit Zufallszahlen erzeugen *)
        FOR i := 1 TO n DO
            data[i] := Random(1000000) ;
        END ;

        (* Laufzeitmessung fuer unterschiedliche Werte von k *)
        FOR k := 0 TO 150 BY 5 DO
            (* erzeuge Arbeitskopie des Feldes, damit jeder Aufruf die
               gleichen Eingabedaten hat *)
            work := data ;

            t1 := Tick.Now() ;
            CombinedSearch(work, k) ;
            t2 := Tick.Now() ;

            IO.PutInt(k) ; IO.Put("\t") ;
            IO.PutReal(FLOAT(Tick.ToSeconds(t2 - t1))) ; IO.Put("\n") ;
            times[k] := times[k] + FLOAT(Tick.ToSeconds(t2 - t1)) ;
        END ;
        IO.Put("\n") ;
    END ;

    (* Mittelwerte ausgeben *)
    IO.Put("# averages:\n") ;
    FOR k := 0 TO 150 BY 5 DO
        IO.PutInt(k) ;
        IO.Put("\t") ;
        IO.PutReal(times[k] / FLOAT(num_tests)) ;
        IO.Put("\n") ;
    END ;
END Main .
```